AMENDMENTS TO THE SPECIFICATION:

Please amend the specification as follows:

On page 1, after the title and before line 4, please add the following paragraph:

This application is a national stage filing under 35 U.S.C. § 371 of International Application No. PCT/EP03/009833, filed on September 4, 2003, which published in the English language, and claims the benefit of priority to U.S. Provisional Application Nos. 60/408,901, filed on September 9, 2002, 60/408,902, filed on September 9, 2002, 60/408,903, filed on September 9, 2002, 60/408,905, filed on September 9, 2002, 60/409,606, filed on September 11, 2002, and 60/409,593, filed on September 11, 2002.

Page 2, please amend the paragraph beginning at line 26, as follows:

As long as the data objects are still available in the original ~~data base~~ database, they can still be modified by any software application during ~~said~~ the time gap. Because the deleting program does not compare the archived data object and the data object to be deleted, such modifications can be lost. This has not only the consequence of the loss of the amended data, it can additionally have the consequence that certain business processes can not be completed.

Page 3, please amend the paragraph beginning at line 1, as follows:

~~An other~~ Another problem arises[[,]] if several archiving processes run in parallel. ~~Then it can happen, that~~ In this scenario, that one data object ~~is~~ can be archived several times, and [[is]] no longer unambiguously identifiable. This can have the consequence

that evaluations or statistical analysis, which use the archive files, produce wrong results.

Page 3, please amend the paragraph beginning at line 8, as follows:

It ~~can also happen~~ is also possible that data objects in the original ~~data base~~ database are read by the writing module and are simultaneously modified by an other software application. In such a case, the data can be transferred from an archiveable status to a non-archiveable status. ~~In consequence~~ As a result, data objects which are not archiveable are written into the archive file and are deleted from the original ~~data base~~ database. In effect, this can result in a loss of data.

Page 3, please amend the paragraph beginning at line 26, as follows:

In accordance with one embodiment of the invention, as embodied and broadly described herein, methods and systems consistent with the principles of the invention provide a data structure for ~~enabling~~ preventing in a computer system ~~an~~ access to a data object having an identifier (ID), comprising:[[

a first lock object, in which the ID of a data object is stored, and in which a link to a storage location of the data object is assigned to ~~said~~ the ID, and

a second lock object in which the ID of the data object is stored,

~~said~~ the lock object~~s~~ being accessible by a software application.

Page 4, please amend the paragraph beginning at line 7, as follows:

By accessing this data structure, software applications, which may require access to data objects, can check by querying the lock objects, whether the data to be

accessed are subject to a moving process or not. If ~~yes~~ the data are subject to a moving process, the access to that data can be postponed until the moving is completed.

Page 4, please amend the paragraph beginning at line 14, as follows:

In accordance with another [[aspect,]] embodiment of the invention, as embodied and broadly described herein, ~~methods and~~ systems consistent with the principles of the invention are provided ~~a computer system for processing data by means of or in a software application for enabling~~ for preventing ~~in a computer system an~~ access to a data object having an identifier (ID), comprising:[[

-]] memory means having program instructions;

[[-]] input means for entering data;

[[-]] storage means for storing data;

[[-]] a processor responsive to the program instructions; and

[[-]] a data structure comprising:[[

]] a first lock object, in which the ID of a data object is stored, and in which a link to a storage location of the data object is assigned to ~~said~~ the ID, and[[

]] a second lock object in which the ID of the data object is stored,[[

said]] the lock objects being accessible by a software application.

Page 5, please amend the paragraph beginning at line 6, as follows:

~~An~~ One advantage of the invention and its embodiments is that the security against data loss in data moving and archiving procedures ~~is~~ may be greatly improved. This ~~avoids in consequence~~ may enable the avoidance of spending a lot of time and/or money for data ~~retrieving~~ retrieval.

Page 6, please amend the paragraph beginning at line 1, as follows:

Fig. 1 is a schematic block diagram of an exemplary ~~the implementation of the inventive data structure within a~~ computer system for implementing methods consistent with the present invention.

Page 6, please amend the paragraph beginning at line 9, as follows:

Fig. 3 is ~~an exemplary~~ a flow diagram of an exemplary implementation of the creation of a first and second lock object shown in Fig. 1.

Page 6, please amend the paragraph beginning at line 13, as follows:

Fig. 4 is ~~an exemplary~~ a flow diagram of an alternative exemplary implementation of the creation of a first and second lock object shown in Fig. 1.

Page 6, please amend the paragraph beginning at line 17, as follows:

Fig. 5 is ~~an exemplary~~ a flow diagram of an exemplary implementation of a deleting process in the context of data moving/archiving.

Page 6, please amend the paragraph beginning at line 21, as follows:

Fig. 6 is ~~an exemplary~~ a flow chart of a further exemplary implementation of the creation of a first and second lock object shown in Fig. 1.

Page 6, please amend the paragraph beginning at line 25, as follows:

Fig. 7 ~~shows an exemplary~~ is a flow chart of an exemplary method to demonstrate how any software application ~~can~~ may use the inventive concept of the first and second lock object.

Page 6, please amend the paragraph beginning at line 29, as follows:

Fig. 8 ~~shows a process alternative to that shown in Fig. 7~~ is a flow chart of an exemplary method to demonstrate how any software application may use the concept of the first and second lock object, including a conditional deletion of a P-lock.

Page 6, please amend the paragraph beginning at line 32, as follows:

Fig. 9 ~~shows an example of~~ is a flow chart for an exemplary method for implementation by a software module ~~by means of~~ through which the locks can be deleted.

Page 7, please amend the paragraph beginning at line 2, as follows:

Computer ~~system~~ systems and ~~program~~ programs are closely related. As used hereinafter, phrases, such as "the computer provides," ~~and~~ "the program provides or performs specific actions[[",]]," and "a user performs a specific action" are convenient ~~abbreviation~~ abbreviations to express actions by a computer system that is controlled by a program or to express that the program or program module is designed to enable the computer system to perform the specific action or the enable a user to perform the specific action by means of a computer system.

Page 8, please amend the paragraph beginning at line 20, as follows:

An identifier (ID) is a type of data, which allows an unambiguous identification of the data object to be archived[[,]]; [[it]] It can be implemented for example as a number or a combination of alphanumerical characters or as a characteristic part of the data object to be archived. It is clear from that definition that a data object can have a wide

variety of IDs. A lock object is a data object, in which the identifiers are stored. It can be implemented, e.g., as a file on a storage means or as a data array in computer memory. The first lock object can be stored advantageously in a nonvolatile storage means and the second lock object can be stored in volatile and/or nonvolatile storage means.

Page 9, please amend the paragraph beginning at line 6, as follows:

Fig. 1 ~~depicts one example~~ is a schematic block diagram of an exemplary implementation of a ~~first embodiment of the invention~~ computer system. Fig. 1 shows a computer system 101 comprising a computer 103 having a CPU 105, a working storage 112, in which an software application 111 is stored for ~~being processed~~ processing by CPU 105. Further, the second lock object 115 is stored in working storage 112 as well, whereas the first lock object is stored on a first storage means 107. Software application 111 comprises program modules 106, 109, 110 for carrying out reading access, writing access and for checking whether the IDs of the selected data objects are contained in the first and second lock objects. Computer [[S]]system 101 further comprises input means 113, output means 112 for interaction with a user, and general input/output means 104, including a net connection 114, for sending and receiving data. A plurality of computer systems 101 can be connected via the net connection 114 in the form of a network 113. In this case, the network computers 113 can be used as further input/output means, including the use as further storage locations. Computer system 103 further comprises the first storage means 107, in which the data objects and the first lock object are stored. A second storage means 108, is available for archiving purpose.

Page 10, please amend the paragraph beginning at line 15, as follows:

In a second implementation of the invention, a data object ~~comprises~~ may comprise one or more fields of one or more tables, and the ID of the respective object ~~comprises~~ may comprise one or more key fields of that data object. This can be seen from Fig. 2. In this instance, various sets of data objects are created in the form of two-dimensional data arrays, i.e. two tables having columns named field A to field X and field A to field Y, respectively, and a certain, unspecified number of lines. A field of the array or table is defined by the name of the column and the respective line. Such field can contain data to be archived. It can alternatively contain a reference to a line of a further table. For example, in table 1 field X in line 2 contains a reference to line 3 in table 2. A data object to be archived comprises of fields of one line of the respective table. If one of the fields contains a reference to a line of an other table, fields of this referenced line belong to the data object, ~~too~~ as well. In the example in Fig. 2, a data object to be archived comprises the fields of line 2 in table 1 and fields of line 3 in table 2.

Page 11, please amend the paragraph beginning at line 1, as follows:

An ID of such a data object can be implemented by the content of one or more so-called key fields, if the combination of these key fields is unique within the respective table. In the example, the fields of "field A" and "field B" can be used as key fields for table 1, whereas field A alone is key field in table 2. Within this example, the data object has the content of the fields of columns field A and B of the respective lines as ID. The ID for the data object to be archived is stored as a first type ID in a first type lock object,

named persistent lock object in Fig. 2, and as a second type ID in a second type lock object, named transactional lock object. The persistent lock object ~~is~~ may be implemented as a table having two columns, the first of which contains the first type ID 1. The second type ID, ID 2, can be implemented as a data array, for example, as a one-dimensional data array stored in the working memory of the computer system. However, it can be implemented as a file on a nonvolatile storage means, too. The first type ID, ID 1, is deleted in a moving or archiving process after the selected data object has been deleted from its original storage location. The second type ID, ID 2, is deleted immediately after a read or write access on a data object has been completed. Alternatively, type ID 1 IDs can be deleted after all the selected data objects have been deleted from the original storage location. As can be seen, both ID types have identical content, the ID of the respective lines of the data to be moved/archived. The persistent lock objects further contain a column by which a filename [[is]] may be assigned to the ID of the data object, i.e. that data object to be archived. In the example, line 1 is archived in a file named 001, lines 2 and 3 in file 002, and line 4 in file 003.

Page 12, please amend the paragraph beginning at line 11, as follows:

A further embodiment ~~is characterized in that said~~ may comprise of first and second lock objects that are created by an data moving or data archiving process.

Page 12, please amend the paragraph beginning at line 15, as follows:

In order to better understand the inventive data structure and its advantages, the creation of the lock objects is now described in more detail with reference to Figs. 3 to 5, which are schematic flow diagrams of exemplary methods that may be implemented

by the ~~implementations of a~~ data moving or archiving processes, respectively, as shown in Fig. 1. Within the context of this description, and particularly with respect to Figs. 3 to 9, a first type ID is called a P-lock (permanent) and a second type ID is called a T-lock (transactional). ~~So~~ Therefore, setting a P- or T-lock for a selected object means to store an ID of that object in a respective lock object. The term "permanent" results for the property of the P-lock of existing permanently, as long as the data object is not yet deleted from its original storage location. The term "transactional" results from the property of the T-lock of existing only as long as a specific action (e.g., checking of archiveability) is performed on a selected data object or, in other words, of being deleted ~~shortly~~ after the respective action has been performed.

Page 13, please amend the paragraph beginning at line 1, as follows:

In the exemplary flow chart of the selecting module in Fig. 3, a data object is selected in a first step 301. Subsequently, a T-lock is set on this object in step 302. If the T-lock was successfully set (step 303), that is, if it did not yet exist, it is checked in step 304 whether a P-lock already exists in the selected data object. If not, the next data object is selected in step 309. The setting of the T-lock (step 302) and the check (step 303), whether it is successfully set, can advantageously be implemented as one "atomic" step. This means that both steps can be executed essentially at the same time or, in other words, the time gap between both steps can be essentially zero.

Page 13, please amend the paragraph beginning at line 15, as follows:

Both checks (steps 303 and 304) can also be implemented by querying the respective lock objects. If a P-lock exists, the T-lock is deleted (step 308) and the next

data object is selected (step 309). If no P-lock exists, it is checked in steps 305 and 306[[,]] whether the data object is archiveable. Such checking comprises a test of whether the data in the data object is readable, complete, or not fraught with obvious failures, etc. If the test is successful, a P-lock is set on that data object in step 307, whereby no archive file is assigned to the data object at that point. Then the T-lock is deleted (step 308) and the next data object is selected (step 309).

Page 13, please amend the paragraph beginning at line 29, as follows:

In the exemplary flow chart of the writing module in Fig. 4, a data object is selected in a first step 401. Subsequently, a T-lock is set on this object in step 402. If the T-lock was successfully set (step 403), it is checked in step 404 whether a P-lock already exists in the selected data object, whereby no file must be assigned to that data object at that point of the process. If the condition is not fulfilled, the T-lock is deleted in step 407, and the next data object is selected in step 408. If a P-lock exists, the data object is stored in an archive file in step 405 and the archive file is assigned to the data object in step 406, e.g., by adding the file name to the lock object as shown in Fig. 2. Subsequently, the T-lock is deleted (step 407), and the next data object is selected (step 408).

Page 14, please amend the paragraph beginning at line 11, as follows:

In the exemplary flow chart of the deleting module in Fig. 5, a data object that has already been archived is selected (step 501). This can be implemented by checking the archive files. If a data object has been selected and successfully read from the

archive file, that data object is deleted from the original storage location (step 502), the P-lock is deleted (step 503), and the next data object is selected (step 504).

Page 14, please amend the paragraph beginning at line 20, as follows:

In the exemplary flow chart of a further exemplary implementation of the creation of a lock object in Fig. 6, the processes, as described above with respect to Figs. 3 and 4, are combined to one module. Accordingly, a data object is selected in a first step 601. Subsequently, a T-lock is set on this object in step 602. If the T-lock was successfully set (step 603), it is checked in step 604 whether a P-lock already exists in the selected data object. If the T-lock was not successfully set, the next data object is selected (step 610). If a P-lock exists on that object, the T-lock is deleted (step 609) and the next data object is selected (step 610). If no P-lock exists on that object, it is checked in step 605[[,]] whether the data object is archiveable. If this check fails (step 606), the T-lock is deleted (step 609), and the next data object is selected (step 610). If the check is positive, the data object is stored (step 605) in an archive file, a P-lock is set (step 608) with the archive file assigned, the T-lock is deleted (step 609), and the next data object is selected (step 610).

Page 15, please amend the paragraph beginning at line 7, as follows:

Fig. 7 shows ~~by way of an exemplary~~ a flow chart of an exemplary method to demonstrate how any software application according to the invention can use the inventive concept of the P[[-]] and T-locks to ensure that the measures, that the software application is going to apply on the data object, do not influence the archiving process. A software application which is programmed to have a read and/or write

access to data objects, which can be subject of an archiving process as described, ~~comprises~~ may comprise the following steps as shown in Fig. 7. In a first step 701, the data object is selected. Then a T-lock is set in step 702 on that object by the application. If the T-lock is successfully set (step 703), it is checked in step 704, whether a P-lock exists on that object[[,]]; otherwise the application terminates in step 707. If a P-lock exists on that object (step 704), the T-lock is deleted (step 706), and the application terminates (step 707). If no P-lock exists, i.e., the data object is not subject to an archiving process, the application can have read/write access to the data object in a working step 705. Subsequently the application deletes the T-lock (step 706) and terminates (step 707).

Page 15, please amend the paragraph beginning at line 30, as follows:

Fig. 8 ~~shows a process alternative to that shown in Fig. 7~~ is a flow chart of another exemplary method to demonstrate how any software application may use the concept of the first and second lock objects, including a conditional deletion of a P-lock. In a first step 801, the data object is selected. Then, a T-lock is set on that object by the application (step 802). If the T-lock is successfully set (step 803), it is checked (step 804), whether a P-lock exists on that object[[,]]; otherwise the application terminates (step 809). If no P-lock exists (step 804), i.e., the data object is not subject to an archiving process, the application can have read/write access to the data object in working step 807. Subsequently, the application deletes the T-lock (step 808) and terminates (step 809). If a P-Lock exists 804, it is checked 805[[,]] whether a file is assigned to it. If a file is assigned, the application deletes the T-lock (step 808) and

terminates (step 809). If no file is assigned, the P-lock is deleted (step 806), and the application can have read/write access to the data object (step 807). Subsequently, the application deletes the T-lock (step 808) and terminates (step 809).

Page 16, please amend the paragraph beginning at line 21, as follows:

Fig. 9 ~~shows an example of~~ is a flow chart ~~for~~ of an exemplary method for implementation by a software module ~~by means of~~ through which the locks set by the modules described above can be deleted. This can be useful in cases in which no archive files are assigned to P-locks or in which P-locks have been deleted for a user. Therein, a P-lock is nothing else than a data object and can be treated in the same way as described above. In a first step 901, a P-lock is selected. Then, a T-lock is set to the P-lock in step 902. If the T-lock is successfully set (step 903), it is checked in step 904, whether the P-lock has a file assigned. If the T-lock is not set successfully, the module terminates (step 907). If the selected P-lock has no file assigned (step 904), the P-lock is deleted (step 905). Then, the T-lock is deleted (step 906), and the module terminates (step 907). Alternative to the termination step 907, a next P-lock can be selected.